Introduction to the Monte Carlo Method

Ryan Godwin

ECON 7010

The Monte Carlo method provides a laboratory in which the properties of estimators and tests can be explored. Although the Monte Carlo method is older than the computer, it is associated with repetitive calculations and random number generation, which is greatly assisted by computers.

The Monte Carlo method was used as early as 1933 by Enrico Fermi, and likely contributed to the work that won him the Nobel Prize in 1938 (Anderson, 1986, p. 99). The term "Monte Carlo" was coined in 1947 by Stanislaw Ulam, Nicolas Metropolis, and John von Neumann, and refers to Stanislaw Ulam's gambling uncle (Metropolis, 1987). The spirit of the method is well captured in the sentiments of Stanislaw Ulam, as he recalls his first thoughts and attempts at practising the method. He was trying to determine the chances that a hand of solitaire would come out successfully. He wondered if the most practical method would be to deal one-hundred hands, and simply observe the outcome (Eckhardt, 1987).

The use of random generation and repetitive calculation are the two central tenets to Monte Carlo experimentation, a method which has flourished since the first electronic computer was built in 1945, and a method which has had a profound impact on mathematics and statistics.

> "At long last, mathematics achieved a certain parity - the twofold aspect of experiment and theory - that all other sciences enjoy" (Metropolis, 1987, p.130).

A Simple Monte Carlo Experiment

We have seen the derivation of some of the properties of the OLS estimator $\boldsymbol{b}$ for $\boldsymbol{\beta}$, in the simple linear regression model. Namely, we have seen that OLS is unbiased and efficient. What if we could *observe* these properties? One of the uses of the Monte Carlo method is to guess at the properties of statistics; properties which may be difficult to derive theoretically.

Let's consider the *unbiasedness* property of OLS, which says that the mean of the *sampling distribution* of $\boldsymbol{b}$ is $\boldsymbol{\beta}$. If we could mimic the sampling distribution, we should be able to observe this property. Recall how we interpreted the sampling distribution:

1. Repeatedly draw all possible samples of size *n*.
2. Calculate values of $\boldsymbol{b}$ each time.
3. Construct a relative frequency distribution for $\boldsymbol{b}$.

If we replace "all possible" in Step #1 with "10,000" or "100,000", then the sampling distribution may easily be synthesized using a computer.

Our Monte Carlo experiment will begin by pretending that the true unobservable population model is *known*. Then, when we calculate the OLS estimates, we pretend that the population model is *unknown*. This way, we can compare $\boldsymbol{b}$ to $\boldsymbol{\beta}$. Let's begin with an overview of the experiment before writing computer code:

1. Specify the (unobservable) population model: $\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon}$. This involves choosing values for $\boldsymbol{\beta}$, choosing the distribution for $\boldsymbol{\varepsilon}$, and creating some arbitrary X data (which will be fixed in repeated samples, in accordance with assumption A.5).
2. "Draw" a sample of $\boldsymbol{y}$ from the population model. This involves using a *random number generator* to create the $\boldsymbol{\varepsilon}$ values.
3. Calculate $\boldsymbol{b}$.
4. Repeat the above steps many (10,000) times, storing each $\boldsymbol{b}$.
5. Take the average of all 10,000 $\boldsymbol{b}$. If $\boldsymbol{b}$ is unbiased, this average should be close to $\boldsymbol{\beta}$.

R Code

First we need to determine some parameters for our experiment, such as sample size and the number of Monte Carlo repitions we would like to use.

```
n = 50
rep = 10000
```

Next, choose values for $\boldsymbol{\beta}$. We'll just use an intercept and single x variable.

```
beta1 = 0.5
beta2 = 2
```

We also need to create the x variable. To do this, we will create n random numbers from the uniform (0,1) distribution.

```
x = runif(n)
```

Take a look at x:

```
 > x
 [1] 0.16113175 0.95362471 0.47450286 0.89152313 0.12962974 0.01736195
 [7] 0.45421529 0.75819744 0.09663753 0.51222232 0.47904268 0.93851048
[13] 0.57261715 0.22855245 0.42623832 0.69128449 0.91723239 0.86308324
[19] 0.83708109 0.70848409 0.02601843 0.38442663 0.23403509 0.80584167
[25] 0.70558551 0.54727753 0.98413499 0.63819489 0.21897050 0.98055095
[31] 0.69164831 0.32517447 0.36495332 0.90024951 0.54707758 0.92455957
[37] 0.41021164 0.99205363 0.40688771 0.11455678 0.98368243 0.06997619
[43] 0.85802275 0.58978543 0.13004962 0.45697634 0.12341920 0.62945295
[49] 0.67256565 0.63599985
```

Let's create vectors of zeros, that will later be used to store the OLS estimates for the intercept and slope:

```
b1 = b2 = rep(0,n)
```

Start the Monte Carlo loop:

```
for(j in 1:rep){
```

Create the disturbances vector (***ε***) by randomly generating numbers from the N(0,1) distribution.

```
eps = rnorm(n)
```

Now "draw" a random sample of y values:

```
y = beta1 + beta2*x + eps
```

Calculate and record OLS estimates, and end the Monte Carlo loop:

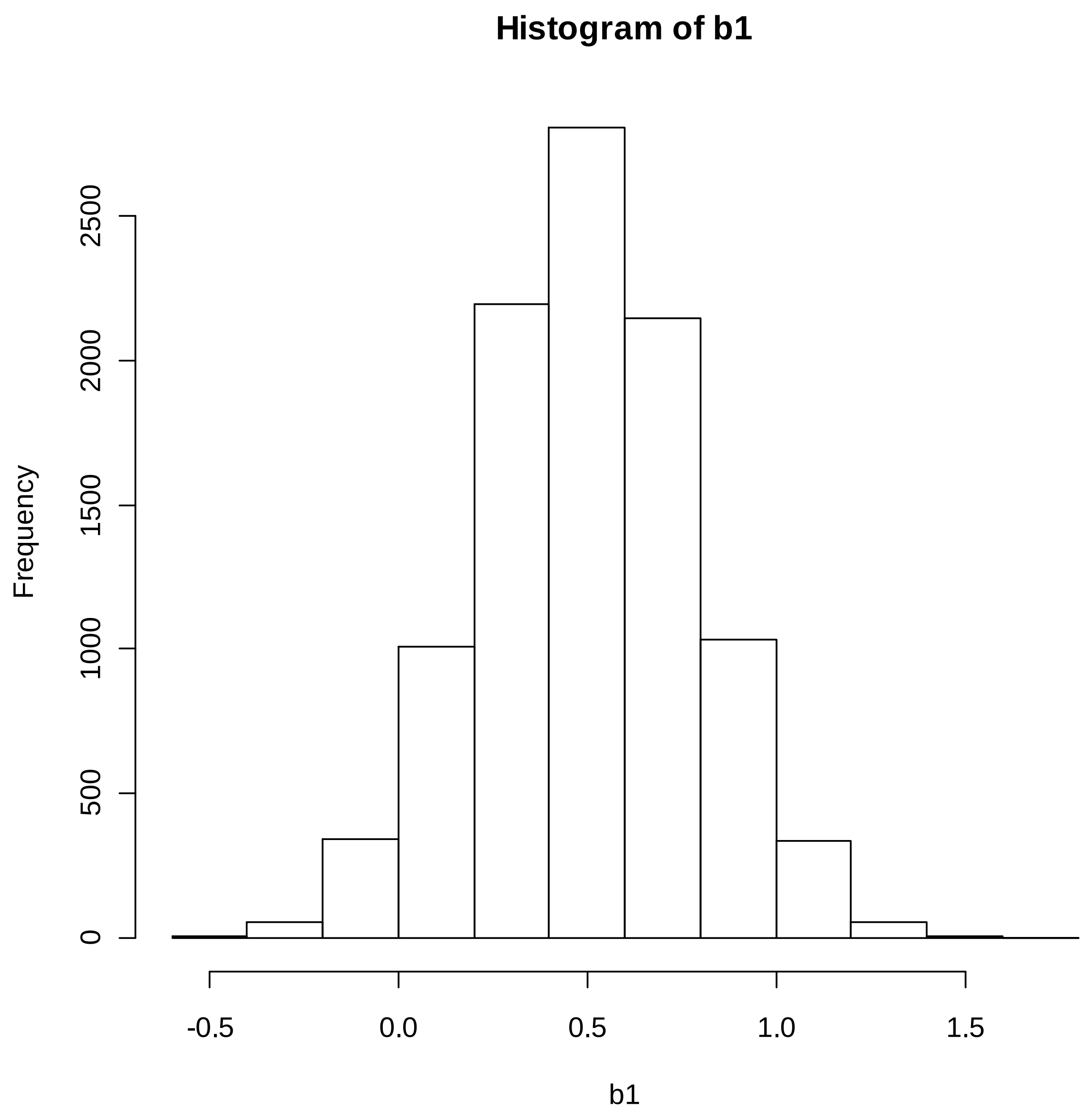
```
b1[j] = lm(y~x)$coefficient[1]
b2[j] = lm(y~x)$coefficient[2]
}
```

Now, we let's see if `b2` appears to be unbiased:

```
> mean(b2)
[1] 2.003847
```

We can also take a look at the simulated sampling distribution:

```
> hist(b1)
```



References

Anderson, H.L. (1986). Metropolis, Monte Carlo, and the MANIAC. *Los Alamos Science* 14: 96-107.

Metropolis, N. (1987). The beginning of the Monte Carlo method. *Los Alamos Science* 15: 125-130.

Eckhardt, R. (1987). Stan Ulam, John von Neumann, and the Monte Carlo method. *Los Alamos Science* 15: 131-137.